Reliable Packets
User-level Mobile Communication
Using Packet Filters

Victor C. Zandy
zandy@cs.wisc.edu
University of Wisconsin - Madison
Motivation

Network connections operate in hazardous conditions

- Flaky Links
- Batteries
- DHCP
- NAT

- Host Movement
- Process Migration
- Hardware Swapping
- ISP Bankruptcy
Network Connection Reliability

• Protect network connections of ordinary applications from
  • IP address changes
  • Link failures
  • Extended periods of disconnection
• Automatically detect failures and recover
• Preserve in-flight data
• User-level implementation
Issues

• Application Transparency
• Interoperability
• Process Checkpointing and Migration
The Scenario

Local Host

User

Application

Sockets API

Kernel

TCP Socket

system calls

packets

Peer Host

User

Application

Sockets API

Kernel

TCP Socket

system calls

packets

Network

Reliable Packets
Reliable Sockets (Rocks) Approach

Interpose rocks library between application and kernel

- Application sees sockets API
- Rocks library hides failures
Reliable Sockets (Rocks) Approach

Application transparency is cumbersome
- Sharing state (fork)
- Interposing on static calls
- Additional interposed libraries
Reliable Packets (Racks) Approach

- Match and operate on packets as they traverse the network stack
- Redirect to user space

Diagram:
- User
  - Application
  - Sockets API
- Kernel
  - TCP Socket
  - Packet Filter
  - Network
Reliable Packets (Racks) Approach

Rackd interposed in packet flow
- Detects connection failures
- Preserves local socket
- Handles reconnection
- Application runs no extra code
Racks: Connection Establishment

- Intercept outbound SYN
- Detect remote functionality
- Initialize with remote rackd or rock library
- Complete connection
Racks: Failure Detection

- Periodically transmit TCP Keep Alive Probe (KAP)
- TCP socket responds with ACK
- Does not interfere with data packets
Racks: Connection Suspension

- Send zero-window advertisements to TCP socket
- TCP socket never gives up
Reconnection

Host A

Host B

Reliable Packets
Reconnection

Host A Changes IP Address

Network
Reconnection

Each end attempts to reconnect to its peer at its last known address.
Racks: Reconnection

- After establishing a new connection, splice the new connection with the old one.
Interoperability

How can racks/rocks tell when they are connected to another rack/rock?

• Must not interfere with ordinary sockets
Enhancement Detection Protocol

• Detects remote rock or rack at connection time
• Works in both directions
• Based on common user-level functionality
• Generally useful for other socket enhancements

• Increases connection time
  • But servers are not penalized
• Can fail in rare cases
Enhancement Detection Protocol

Rack/Rock Client
connect

TCP Handshake

Rack/Rock Server
accept

Reliable Packets
Enhancement Detection Protocol

Rack/Rock Client
  connect
  shutdown

TCP Handshake

Rack/Rock Server
  accept
  read

EOF

Suspects client is enhanced
Enhancement Detection Protocol

Rack/Rock Client
- connect
- shutdown
- read
- close

Rack/Rock Server
- accept
- read
- write
- close

TCP Handshake
- EOF

"I’m Enhanced!"

Knows server is enhanced

Suspects client is enhanced
Enhancement Detection Protocol

Rack/Rock Client
  connect
  shutdown
  read
  close
  connect

TCP Handshake

Rack/Rock Server
  accept
  read
  write
  close

EOF

“I’m Enhanced!”

Knows server is enhanced

Suspects client is enhanced
Enhancement Detection Protocol

Rack/Rock Client
- connect
- shutdown
- read
- close
- connect
- write

TCP Handshake

Rack/Rock Server
- accept
- read
- close
- accept
- read

"I’m Enhanced!"

Suspects client is enhanced

Knows server is enhanced

Knows client is enhanced
Enhancement Detection Protocol

Rack/Rock Client
- connect
- shutdown
- read
- close
- connect
- write

TCP Handshake

EOF

“I’m Enhanced!”

Rack/Rock Server
- accept
- read
- write
- close
- accept
- read

Suspects client is enhanced

Knows server is enhanced

Knows client is enhanced

Negotiate, Initialize, Return to Application

- 24 - Reliable Packets
ProcessCheckpointing

• Racks and rocks can checkpoint programs that communicate over sockets
  • Condor jobs
  • MPI, PVM jobs
  • Desktop applications
• Racks/Rocks reconnection mechanisms
  • Recover kernel-level state
  • Recover in-flight messages
Rack Checkpoints

Rack complications:
- Checkpoint kernel socket buffers
- Restore process socket
Rock Checkpoints

- Rocks checkpoint naturally with process address space
- Checkpoint *vanilla* Condor jobs with open connections
- Checkpoint MPI jobs

User
- Application
- Ckpt Library
- Rocks Library
- Sockets API

Kernel
- TCP Socket

Network
Performance

- High connection setup overhead (20x)
- Enhancement detection protocol
- Authentication initialization
Conclusion

- Packet filters for mobile network connections
  - Claim advantages of kernel-level designs
  - Avoid transparency problems of bypassing
- New enhancement detection protocol
- Come see the demos!
  - Mobile computing
  - Process migration
- Download now (x86 Linux)
  - http://www.cs.wisc.edu/~zandy/rocks
UDP

• Fundamentally different from TCP
  • No connection model
  • Application defines reliability model

• Rocks and racks can provide critical information to mobile-aware applications
  • RE-API conveys disconnection notification and reconnection information

• For a few UDP applications, racks and rocks can usefully provide reliability
Related Work

• Mobile IP, TCP Migrate, MSOCKS
• Emphasize mobility, not reliability
  • No extended periods of disconnection
  • Lack mechanisms for failure detection and automatic reconnection
• Based on kernel modifications
  • Privileged, unportable
  • Based on implementation, rather than interface and protocol